Teresa M. Corbin (SBN 132360)
Denise M. De Mory (SBN 168076)
Jaclyn C. Fink (SBN 217913)
HOWREY LLP
525 Market Street, Suite 3600
San Francisco, California 94105
Telephone: (415) 848-4900
Facsimile: (415) 848-4999

Attorneys for Plaintiff SYNOPSYS, INC. and for Defendants AEROFLEX INCORPORATED, AMI SEMICONDUCTOR, INC., MATROX ELECTRONIC SYSTEMS, LTD., MATROX GRAPHICS, INC., MATROX INTERNATIONAL CORP., MATROX TECH, INC., and AEROFLEX COLORADO SPRINGS, INC.

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

SAN FRANCISCO DIVISION

RICOH COMPANY, LTD.,

Plaintiff,

vs.

AEROFLEX INCORPORATED, AMI SEMICONDUCTOR, INC., MATROX ELECTRONIC SYSTEMS LTD., MATROX GRAPHICS INC., MATROX INTERNATIONAL CORP., MATROX TECH, INC., AND AEROFLEX COLORADO SPRINGS, INC.

Defendants.

SYNOPSYS, INC.,

Plaintiff,

vs.

RICOH COMPANY, LTD.,

Defendant.

Case No. C03-04669 MJJ (EMC)

Case No. C03-02289 MJJ (EMC)

NOTICE OF MOTION AND MOTION FOR SUMMARY JUDGMENT OF INVALIDITY OF CLAIMS 13-17 OF U.S. PATENT NO. 4,922,432

Dispositive Motion No. 4

Date: September 26, 2006
Time: 9:30 a.m.
Courtroom: 11, 19th Floor
Judge: Martin J. Jenkins

TABLE OF CONTENTS

TABLE OF AUTHORITIES

Page(s)

CASES

STATUTES

RULES

PLEASE TAKE NOTICE that on September 26, 2006, at 9:30 a.m., before the Honorable Martin J. Jenkins in Courtroom 11, 19th Floor, in the United States District Court, 450 Golden Gate Avenue, San Francisco, California, Plaintiff Synopsys, Inc. ("Synopsys") and Defendants Aeroflex Incorporated, Aeroflex Colorado Springs, Inc., AMI Semiconductor, Inc., Matrox Electronic Systems Ltd., Matrox Graphics Inc., Matrox International Corp., and Matrox Tech, Inc. ("the Customer Defendants") will move for summary judgment pursuant to Rule 56 of the Federal Rules of Civil Procedure that claims 13-17 of the patent-in-suit, U.S. Patent No. 4,922,432 (the "'432 patent") is invalid for failure to comply with the statutory requirements of 35 U.S.C. §§ 102, 103, and 112.

This motion is based on the memorandum of points and authorities set forth below, the accompanying declarations, exhibits, and proposed order, any argument of counsel at the hearing on this motion, and all other pleadings and matters of record in these actions.

I. INTRODUCTION

It is axiomatic that, in order to be valid, a patent must be novel and nonobvious. That is, a patent must be a true invention, not just a retread of what someone else previously created, or such a small variant from what preceded it that any person of skill in the art would have been able to create it from the prior art. Unfortunately for Ricoh, that is precisely the case with the '432 patent. Claims 13-17 — the only asserted claims — never should have issued. The "invention" described in these claims was entirely anticipated (or, at the least, rendered obvious) by myriad references published more than one year prior to the January 13, 1988 filing date of the patent. These facts invalidate Claims 13-17 of the patent, and Defendants seek an order granting them summary judgment of invalidity of the '432 patent.

Claims 15 and 17 are invalid for the additional reason that they do not claim any further limitation beyond their parent claim. This is a violation of § 112 ¶ 4, and requires the invalidation of these claims.

II. STATEMENT OF FACTS

Although there are many prior art references that anticipate the '432 patent, to simplify this motion, Defendants move only on the Kowalski/VDAA references. Below, we present the basic elements of the asserted claims of the '432 patent, and the Kowalski/VDAA prior art.

A. **The invention described in Claims 13-17 of the '432 patent.**

The '432 patent issued on May 1, 1990 out of an application filed on January 13, 1988. As the Court is aware, the '432 patent describes a synthesis tool — that is, a computer program that takes an architecture independent description of the functionality of an integrated circuit and transforms that description into a gate-level netlist. Although Ricoh has continually trumpeted in this litigation its "invention," there is absolutely no allegation in this case (or could there be) that Ricoh's '432 patent is a pioneer invention — i.e., the first ever synthesis tool. Indeed, Ricoh has stated that the only alleged inventive aspect in the '432 patent is the "unique combination" of elements. Thus, the elements and their flow is critical to understanding the alleged invention and determining whether it was anticipated.

There are six elements in independent claim 13, designated below by letters:

> A computer-aided design process for designing an application specific integrated circuit which will perform a desired function comprising:
> [A] storing a set of definitions of architecture independent actions and conditions;
> [B] storing data describing a set of available integrated circuit hardware cells for performing the actions and conditions defined in the stored set;
> [C] storing in an expert system knowledge base a set of a rules for selecting hardware cells to perform the actions and conditions;
> [D] describing for a proposed application specific integrated circuit a series of architecture independent actions conditions;
> [E] specifying for each described action and condition of the series of one of said stored definitions which corresponds to the desired action or condition to be performed; and
> [F] selecting from said stored data for each of the specified definitions a corresponding integrated circuit hardware cell for performing the desired function of the application specific integrated circuit, said step of selecting a hardware cell comprising applying to the specified definition of the action or condition to be performed, a set of cell selection rules stored in said expert system knowledge base and generating for the selected integrated circuit hardware cells, a netlist defining the hardware cells which are needed to perform the desired function of the integrated circuit and the interconnection requirements therefor.

The six steps in Claim 13 describe a fairly simple process. First, elements A, B, and C are stored to be available for use later. In element D, the ASIC is described in a series of architecture independent actions and conditions, element E takes the described actions and conditions, and specifies a definition for each input action or condition from the definitions stored in element A. In element F, cell selection rules are applied to the output of Element E (the specified definitions) to select hardware

cells contained in element B and a netlist is generated. A very simplified flow of the claimed process is thus:

Store: Defintions Data describing hardware cells Cell election rules
Describe ASIC using a series of actions and conditions
Specify a stored definition for each action and condition
Select stored cells by applying stored cell selection rules to stored specified definitions and generate a netlist of hardware cells and interconnections

Independent Claims 14-17 each purport to add an additional step to the above process. Claim 14 adds the step of generating mask data. Claim 15 adds the step of generating data paths, whereas Claim 16 adds the step of generating these data paths through rules stored in a knowledge base. Claim 17 adds the step of generating control paths.

B. **Claim construction of terms.**

The Court has construed many of the terms used in the patent. Ex. 8.[1] This construction is, of course, what the attached claim charts are based upon. However, the Court has not construed the term

[1] Unless otherwise noted, all exhibits referenced in this motion are attached to the Declaration of Denise M. De Mory In Support of Synopsys' and Customer Defendants' Summary Judgment Motions filed concurrently herewith. All deposition references are likewise included in the De Mory Declaration.

"hardware cells." As Synopsys and the Customer Defendants have explained elsewhere (*see* Summary Judgment Motion No. 2), the plain meaning of "hardware cell" is a cell that maps or corresponds to a specified function from element E. In words, if the definition specified in element e is "add" then the selected hardware cell must be an implementation of an adder, and not as Ricoh contends, primitive logic gates, such as AND, OR, NOR (as Ricoh claims) For purposes of this motion, Synopsys and the Customer Defendants utilize the same definition of "hardware cells" that they do in their non-infringement motion, and defer largely to the explanation set forth in that motion.

The Defendants asserted during claim construction that the term "netlist" in claim 13 should be construed as including "the necessary control and data path information for connecting the hardware cells and the controller" based on the '432 patent's specification and the embodiment disclosed. Ex. 34, Exh. A at 19 (clause "P"). Ricoh argued that Defendants' construction was too narrow, and the Court agreed, defining the phrase "a netlist defining the hardware cells which are needed to perform the desired function of the integrated circuit" as "a description of the hardware components (and their interconnections) needed to manufacture the ASIC as used by subsequent process, e.g., mask development, foundry, etc." Ex.8 at 24. The Court specifically noted that "claim 13 does not *restrict* the interconnection requirements of the hardware cells to 'data and control paths,'" implying that the interconnections the Court was referring to in its construction *does* include data and control paths. *Id.* (emphasis added)

C. **The Kowalski references and the VDAA system.**

In April 1984, a Ph.D. student at Carnegie Mellon University, Thaddeus J. Kowalski, wrote his dissertation on a synthesis tool he dubbed the VLSI Design Automation Assistant ("VDAA").[2] Ex. 35. ("Kowalski Thesis"). Dr. Kowalski published several papers on his VDAA program, including a short article in August 1985 titled "The VLSI Design Automation Assistant: From Algorithms to Silicon." Ex. 36 ("Kowalski85").

[2] Although Dr. Kowalski referred to the system as "DAA," we refer to it as "VDAA" to differentiate this system from a completely different system developed at CMU in the 1970's known as CMU/DA.

As described in detail in the claim charts attached hereto as Exhibits A and B, the Kowalski Thesis and Kowalski85 fully anticipate claims 13-17 of the '432 patent. The VDAA system accepted as input an algorithmic description of the behavior of the chip, written in a language known as ISPS, and the ISPS description described the desired functionality of the chip in terms of actions and conditions. Ex. 35 at 46-47. Thus, element D is met, because the circuit is described.

This ISPS description was then translated into a data-flow graph representation known as VT by the VDAA system. In the process of compiling the design into a VT, the compiler translated each of the actions and conditions into a predefined operator, which forms the node of the graph. Ex. 35 at 49. Thus, both elements A and E are met — the operators are the "definitions of the architecture independent actions and conditions," they were predefined ("stored" as required by element A), and specified (assigned to a stored definition as required by element E).

From there, the nodes in the VT representation were used to select hardware cells from the "technology-sensitive database" using expert rules stored in the VDAA system. *Id.* The rules were in an IF-THEN antecedent format. *Id.* Thus, the stored rules (element C) were used to select the stored hardware cells (element B). After the hardware cells were bound using the module binder, a netlist was created by the control allocator. *Id.* Thus, element F is satisfied.

At deposition, Dr. Kowalski clarified that the "technology sensitive" database discussed in these papers contained cell descriptions that "could be as low as a single an[d] gate or as high and complicated as an ALU." Ex. 37(Kowalski) at 83:11-24.

In addition to his affiliation with CMU, Dr. Kowalski was a researcher at AT&T Bell Laboratories, and after receiving his doctorate in 1984 for his work on VDAA, Dr. Kowalski further refined the program at AT&T. Ex. 37 (Kowalski) at 103:17-104:6. One of these refinements was to eliminate the need for a separate module binder process — the hardware cells were selected, bound, and a netlist was created all in one step. Ex. 37 (Kowalski) at 104:8-25; 117:4-10; 118:7-119:12. This revised VDAA system is described in the claim chart attached hereto as Exhibit C.

Recently, an anonymous requestor asked for the '432 patent to be reexamined by the USPTO in light of the Kowalski Thesis and Kowalski 85. Ex. 38. The PTO granted this request, and the '432 patent is in reexamination on the basis of these two references. Ex. 39. The PTO noted that the

Kowalski references, both individually, together, and in combination with other references, raised substantial new questions of patentability. No office action has yet issued from this reexamination.

III. ARGUMENT

As demonstrated by the attached claim charts, the various Kowalski/VDAA references are anticipatory. In addition, claims 15 and 17 are invalid on a separate basis — for failure to add limitations beyond those of the parent claim.

A. Legal standards.

The Patent Act states that a person is not entitled to a patent if "the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of the application for patent in the United States." 35 U.S.C. § 102 (b). This section is known as a statutory bar, and it absolutely bars the obtaining of a patent on an invention that was (1) publicly used or on sale more the one year prior to the filing date of the patent or (2) disclosed in any prior publication that anticipates the claims or renders them obvious. *See Constant v. Advanced Micro-Devices, Inc.*, 848 F.2d 1560, 1568 (Fed. Cir. 1988) (affirming invalidity of two patents under § 102(b) and describing the section as a "statutory bar via prior description, use, or being on sale.").

Under § 102(b), a device qualifies as prior art if it was in public use more than one year prior to the date of the patent. 35 U.S.C. § 102. An invention is in public use "where it is exposed or demonstrated to persons other than the inventor ... who are under no obligation of secrecy." *Nordberg, Inc. v. Telsmith, Inc.*, 882 F. Supp. 1252, 1283 (E.D. Wis. 1995), *aff'd*, 82 F.3d 394 (Fed. Cir. 1996). "Use of a single specimen, even in a factory and in the presence only of the employees, may be "public" use under certain circumstances." *Id.* at 1284 (citation omitted). Further, a public use "means the dissemination of an idea without substantive restriction." *AMP, Inc. v. Fujitsu Microelectronics, Inc.*, 853 F. supp. 808, 816 (M.D. Pa. 1994) (citation omitted).

When challenging the validity of an issued patent, the moving party must demonstrate invalidity by clear and convincing evidence. *See Enzo Biochem, Inc. v. Gen-Probe, Inc.*, 424 F.3d 1276, 1281 (Fed. Cir. 2005). Despite this standard, summary judgment under the statutory bar is appropriate when there is no material issue of fact that a reference published, used, or sold more than a

year prior to the filing date anticipates or renders obvious a claim. *Telemac Cellular Corp. v. Topp Telecom, Inc.*, 247 F.3d 1316, 1330 (Fed. Cir. 2001) (affirming summary judgment of invalidity of patent in light of anticipating prior publication).

In general, summary judgment is granted to a moving party when "there is no genuine issue as to any material fact" and the "moving party is entitled to judgment as a matter of law." FED. R. CIV. P. 56(c). Summary judgment is just as reasonable in a patent case as in any other case. *See Barmag Barmer Maschinenfabrik AG v. Murata Mach., Ltd.*, 731 F.2d 831, 835 (Fed. Cir. 1984). Although anticipation under § 102(b) is a question of fact, it still may be decided on summary judgment. *Telemac*, 247 F.3d at 1327. Moreover, "[s]ummary judgment is not a disfavored procedural shortcut, but rather an essential thread in the fabric of the Federal Rules that eliminates unfounded claims without recourse to a costly and lengthy trial." *Colgate Palmolive Co. v. W.L. Gore & Assoc., Inc.*, 919 F. Supp. 767, 769 (D.N.J. 1996). Once a party has made an initial showing that summary judgment is warranted, the opposing party may not rest upon pleadings; rather, "the non-moving party must 'designate specific facts showing that there is a genuine issue for trial.'" *Tinoco v. Belshe*, 916 F. Supp. 974, 979 (N.D. Cal. 1995) (quoting *Celotex Corp. v. Catrett*, 477 U.S. 317, 324 (1986)). The Court may grant summary judgment if Ricoh's evidence "is merely colorable, or is not significantly probative." *Tinoco*, 916 F. Supp. at 979 (quoting *Anderson v. Liberty Lobby*, 477 U.S. 242, 249-250 (1986)).

B. **The Kowalski/VDAA references anticipate Claims 13-17.**

The Kowalski Thesis and Kowalski 85 reference are both printed publications dated prior to the critical date of January 13, 1987, with publication dates of 1984 and 1985, as noted on the cover pages of the two references. Exs. 35 and 36. Thus, these two references qualify as potential § 102(b) art. Furthermore, they, along with deposition testimony from Dr. Kowalski, describe a working system which was in public use in the United States more than a year prior to the filing of the patent, so the VDAA system is itself potential § 102(b) art.

Each of these references — the papers and the system itself (in its various versions) — anticipate Claims 13-17 of the '432 patent, as shown in the attached claim charts. As the charts demonstrate, the references describe a synthesis tool which takes as input a behavioral description of a

chip, translates that input using a stored set of definitions, and uses the translated input, a set of stored hardware cells ("modules") and a set of expert rules to select cells and create a netlist. These charts suffice to shoulder the Defendants' burden of proving invalidity by clear and convincing evidence.

1. The references are anticipatory under Synopsys' and the Customer Defendants' definition of "hardware cell."

When properly considered, there is only a very limited dispute about whether or not VDAA anticipates the asserted claims, and that relates solely to the definition of "hardware cell." Dr. Soderman, Ricoh's expert, claims three elements are missing from the Kowalski references and VDAA: the stored hardware cells, the expert rules to select hardware cells, and the netlist of selected hardware cells. These assertions depend entirely on whether or not Ricoh is right on its construction of "hardware cell." Clearly, VDAA uses rules to select modules corresponding to the functions (i.e., specified definitions). This can only fail to meet the claims if Ricoh is correct that rules must be used to select cells consisting solely of primitive gates. Ricoh is not correct.[3]

The language of the first part of the selection step is: "selecting from said stored data [the hardware cell data stored in element B] for each of the specified definitions [specified in element E] **a corresponding integrated circuit hardware cell**." The Court interpreted this language to require "mapping the specified stored function to a corresponding stored hardware cell." Ex. 8 at 20. Synopsys and the Customer Defendants claim that, in accordance with the language of the claim, as well as the Court's clear construction of the claim language, what occurs in this step is that the specified function is being assigned to a "hardware cell."[4] In other words, if the specified definition is "ADD (A,B,C)," which is an exemplary macro in the '432 patent specification, the only type of hardware cell to which the function ADD can be mapped is an adder. Put another way, the only "hardware cell" that corresponds to the function add is an adder. This is the only way that the patent claim makes sense, and this interpretation is consistent with the specification.

---

[3] Synopsys' and the Customer Defendants' full claim construction argument on "hardware cell" is spelled out in Summary Judgment No. 2.

[4] Indeed, the Court's claim construction interprets this element to mean exactly what Synopsys and Customer Defendants contend: "mapping the specified function to a corresponding hardware cell." Claim Construction Order at 20.

The portion of the claim after the comprising step further confirms that the hardware cells need not be individual gates. The claim requires that the selection process be "comprised of" a two step process: (1) "**applying** to **the specified definition** of the action or condition to be performed, **a set of cell selection rules stored** in said expert system knowledge base" and (2) "generating for the selected integrated circuit hardware cells, a netlist defining the hardware cells . . ." In other words, the patent requires that "hardware cells" corresponding to the specified definitions be selected by applying rules to those definitions, and that, ***subsequently***, a netlist be generated "defining" the selected integrated circuit hardware cells.[5] Ex. 1 at 16:61-62. Thus, the first step is applying the rules to the definitions to select the hardware cells, which in the case of VDAA is, as described above, applying the rules to the operators to select the modules. The second step, generating for the selected integrated circuit hardware cells (confirming that the cells were selected in the prior step), a netlist, defining (not comprised of or listing) the hardware cells is likewise practiced by VDAA, occurring during module binding when the actual netlist is generated.

In view of the foregoing, there is no doubt that the VDAA system practices this method via the selection of technology dependent modules by applying rules to the operators in the VT. Specifically, the VDAA, a knowledge based-expert system, applies a set of rules to each of the definitions to select a module. The VDAA modules are "hardware cells," and these modules are selected by rules. Subsequent to that selection, a netlist is generated for these cells in the module binder. Nothing more is necessary, and the Kowalski/VDAA references anticipate the '432 patent. Because Dr. Soderman's and Ricoh's position on this issue is flawed as a matter of law, summary judgment should be granted that the VDAA anticipates claim 13.

---

[5] This also is fully consistent with the patent's specification. The specification says "the netlist is generated after the cells have been selected by the PCSC." Col. 9:64-65. The specification goes on to explain the information that is used in netlist generation which include the information in the cell library, which is comprised of for each hardware cell (also called a block), the following information: type out: Col. 9:24-50. Given that the cell library included transistor level descriptions as well as mask data, a netlist as well as mask data could readily be generated.

2. **The references are anticipatory even under Ricoh's flawed definition of "hardware cell."**

Even if there were a requirement that the "hardware cells" of the patent claim be individual gates (as opposed to representations of circuit components), the original VDAA described in the papers meet that requirement. Dr. Kowalski testified that the "technology sensitive database" had cell descriptions, which *included* single gates as well as more complicated structures such as ALUs. Ex. 37 (Kowalski) at 83:5-24. Thus, the VDAA system, which was in public use more than a year prior to the filing of the patent application, did store individual gates and had those gates selected by expert rules, meeting the "hardware cell" limitations even under Ricoh's definition.

Furthermore, the version of VDAA created by Dr. Kowalski at Bell Labs by 1985 also clearly met Ricoh's requirement that there be direct selection of gates by rules. In the later version of VDAA, created in 1985, Dr. Kowalski incorporated the module binder into the cell selection process. Ex. 37 (Kowalski) at 14:4-16; 17:6-19:7; 125:2-10. This incorporated module binder, called Fred, used the expert rules contained in the VDAA system to choose *and bind* the modules in one step. Ex. 37 (Kowalski) at 116:22-119:9; 127:14-128:6. Even though this version of the VDAA system was never reported via a journal article, the fact is that this version of VDAA existed, and Ricoh cannot seriously dispute that it did exist. It was in public use, because AT&T directed Dr. Kowalski, instead of patenting the system, to "publish widely and place the work in the public domain." Ex. 37 (Kowalski) at 21:6-7. In these circumstances, the version of VDAA created at AT&T was in public use in the United States in 1985, more than one year prior to Ricoh filing of the application for the '432 patent, and is therefore invalidating prior art under § 102(b).

C. **Claims 15 and 17 do not add additional limitations to Claim 13 and are therefore invalid.**

Claim 13 sets forth a clear methodology, as described above. At the end of the process, a netlist is generated. According to the construction given by the Court, this netlist contains all "hardware components (and their interconnections) needed to manufacture the ASIC...." This necessarily means that the netlist includes data paths and control paths, and thus requires that the data paths and control paths be generated prior to the netlist generation claimed in claim 13.

There is one odd thing about this observation. Claims 15 and 17 each purport to add "the further step" of generating data paths (claim 15) and control paths (claim 17) to claim 13. But because claim 13 already includes these steps, dependent claims 15 and 17 are superfluous and add no further limitation to claim 13.[6]

Given this analysis, claims 15 and 17 are invalid because they do not comply with the statutory requirements of 35 U.S.C. § 112, ¶ 4. Section 112, ¶ 4 requires "a claim in dependent form shall ... specify a further limitation of the subject matter claimed." *See Curtiss-Wright Flow Control Corp. v. Velan, Inc.*, 438 F.3d 1374, 1381 (Fed. Cir. 2005). In *Curtiss-Wright*, the Federal Circuit reasoned that "reading an additional limitation from a dependent claim into an independent claim would not only make that additional limitation superfluous, it might render the dependent claim invalid" under 35 U.S.C. § 112, ¶ 4. *Id.* at 1380. The Federal Circuit in *Pfizer, Inc. v. Ranbaxy Labs. Ltd.* recently confirmed the reasoning of the *Curtiss-Wright* opinion when it held a claim invalid for failing to comply with § 112, ¶4. 2006 U.S. App. LEXIS 19416 *19 (Fed. Cir. Aug. 2, 2006). The Federal Circuit properly recognized that it "should not rewrite claims to preserve validity." *Id.* at *18 (citation omitted). "If the only claim construction that is consistent with the claim's language and the written description renders the claim invalid, then ... the claim is simply invalid." *Id.* at *18-19. Because claim 13 implicitly includes the steps of generating data paths and control paths, claims 15 and 17 should be invalidated under § 112, ¶ 4, for failure to recite a further limitation to their parent claim.

IV. CONCLUSION

For the foregoing reasons, Defendants respectfully request that the Court grant their motion and issue a summary judgment of invalidity of the '432 patent.

---

[6] The claim construction doctrine of claim differentiation does not change this result. Claim differentiation merely creates a *presumption* that dependent claims are of different scope than their parent claims; it does not mandate such a result. *Seachange Int'l v. C-COR, Inc.*, 413 F.3d 1361, 1369 (Fed. Cir. 2005) (claim differentiation "presumption is not a hard and fast rule and will be overcome by a contrary construction dictated by the written description or prosecution history") (citation omitted). Where both parties agreed, and the Court found, that the proper construction of the term "netlist" in Claim 13 requires that a netlist contain, as interconnections, (at the very least) data paths and control paths, the presumption has already been overcome.

Dated: August 18, 2006

HOWREY LLP

By: /s/ Denise M. De Mory
Denise M. De Mory
Attorney for Plaintiff SYNOPSYS, INC. and Defendants AEROFLEX INCORPORATED, AMI SEMICONDUCTOR, INC., MATROX ELECTRONIC SYSTEMS, LTD., MATROX GRAPHICS INC., MATROX INTERNATIONAL CORP., MATROX TECH, INC., and AEROFLEX COLORADO SPRINGS, INC.

EXHIBIT A

Thaddeus Julius Kowalski, *The VLSI Design Automation Assistant: A Knowledge-Based Expert System*, Canegie-Mellon University PhD Thesis, April 1984.

13. A computer-aided design process for designing an application specific integrated circuit which will perform a desired function comprising	
storing a set of definitions of architecture independent actions and conditions;	"*The technology-independent-hardware-network level.* The second level is a technology-independent-hardware-network description of the design. At this level of detail, the functionality and connectivity of the hardware can be understood regardless of the target design technology (TTL, ECL, NMOS, or CMOS). This level is represented in the technology-independent-hardware-network language, SCS, which describes technology-independent registers, operators data paths, and control signals. Data-memory allocation and control allocation are defined as the creation of this level from the algorithmic level while applying classical compiler optimizations and design styles." [Pg. 4] "The ISPS description is compiled into a VT data-flow representation, which makes it easier to recognize and implement design decisions by synthesis programs. The VT is a directed acyclic graph, DAG, similar in nature to those used in optimizing compilers, with the addition of control constructs to allow conditionals and subroutines in the VT. The nodes in this graph are called *operators* [i.e. actions] and correspond to operations that take certain values as input and produce new values as output... The arcs connecting the nodes are called *outnodes* and represent the generation or use of data values [i.e. conditions]. They are translations of the ISPS carriers and the temporary carriers needed to pass results from one operator to another. The graph is partitioned into subgraphs called *VT-bodies*, corresponding to a set of operations that can be evoked, entered, or left as a unit. These subgraphs are translations of ISPS procedures, labeled blocks and loops." [pg. 49] "*The working memory components.* The working memory is a collection of elements that describe the current situation. The elements resemble the data structures in conventional programming languages: struct foo { <attribute 1> = <value 1>; ... <attribute n> = <value n>; }" [Pg. 11] "For each design task the DAA is given a working-memory representation of the VT, shown in Tables 6 and 7. The first column provides the name of the attribute, while the second column describes the possible values." [Pg. 52] (See also pg. 59 regarding SCS.) See also Tables 6 and 7. [pg. 50-51] "The technology database provides the name translation between VT and SCS operators, and the number of micro-control steps required for the operation." [pg. 52]
storing data describing a set of available integrated circuit hardware cells for performing the actions and conditions defined in the stored set;	"This technique allows easy change of target technologies and individual fine tuning of design constraints." [Pg. 39] "...the technology database and constraints are initialized as given in Tables 9, 10, 11, and Appendix B." [Pg. 69]

storing in an expert system knowledge base a set of rules for selecting hardware cells to perform the actions and conditions;	"*The rule memory component.* The rule memory is a collection of conditional statements that operate on elements stored in the working memory. The statements resemble the conditional statements of conventional programming languages: *IF: <antecedent 1> ... <antecedent n> THEN: <consequence 1> ... <consequence m>.* [...] The rule memory is a collection of knowledge *chunks* about a particular problem domain. Most rule-based systems contain hundreds of rules that have been *painfully* extracted from months of interviewing experts." [pg. 11. "*The rule memory.* The rule memory is a collection of conditional statements that operate on elements stored in the working memory. The statements resemble the conditional statements of conventional programming languages... Each subtask in the DAA is associated with a set of rules for carrying out the subtask... Most of the rules, like the example above, define situations in which a partial design should be extended in some particular way. These rules enable the DAA to synthesize an acceptable design by determining at each step , whether a certain design extension respects constraints." [Pg. 28] "First, 'book knowledge' of the problem is codified as a set of situation-action rules; interviews with experts then fill in knowledge gaps and refine current knowledge." [Pg. 17] See Table 14. RULES BY FUNCTION AND KNOWLEDGE TYPE [Pg. 63] "The designers [i.e. experts] identified the base-variable storage elements, the database, the constraints, and the controller as global objects that probably would not change and should be allocated first. The base-variable storage elements are memories, registers declared globally across the design, formal parameters, and constraints. [...] This section comprises 78 rules and is done under the context of *declared-variable-allocation.*" [Pg. 68]
describing for a proposed application specific integrated circuit a series of architecture independent actions and conditions;	"*The algorithmic level.* The first level is an algorithmic description of the design. At this level of detail the high-level intent of hardware can be understood and simulated regardless of the target design style (for example pipeline, multiplexer, microprocessor, or bus) or technology. This level is represented in the instruction set processor language, ISPS, and a value trace data and control flow description language, VT." [pg. 3] "An ISPS description consists of a series of declarations of *entities.* Some of these are simple *carriers*, which hold the data being manipulated, similar to variables in a programming language. Other entities are procedures or functions much like the procedures or functions of a programming language. These define how the data is manipulated.. The procedures are specified by data *operators* that do arithmetic, logical, relational, and shift operations on the data, and by sequential, parallel, and conditional constructs that show how the data operators are combined." [Pg. 46-47]
specifying for each described action and condition of the series one of said stored definitions which corresponds	"*The technology-independent-hardware-network level* ... Data-memory allocation and control allocation are defined as the creation of this level from the algorithmic level while applying classical compiler optimizations and design styles." [Pg. 4]

to the desired action or condition to be performed; and	"…the knowledge that expert VLSI designers use to go from the algorithmic level to the technology-independent hardware network level has been extracted. The codified knowledge has been tested for completeness and correctness by implementing an interactive system, DAA, that adds register, operator, data path and control signal detail to the VT description of hardware to form the SCS description." [pg. 5] "The DAA starts with a data flow representation extracted from the algorithmic description… The DAA produces a technology-independent hardware network description. This description is composed of modules, ports, links, and a symbolic microcode. The modules can be registers, operators, memories, and buses or multiplexers with input, output, and bidirectional ports." [Pg. 26] "Next, it maps [i.e. specifies] all data-low operator outputs not bound to base-variable storage elements to register modules. Last, it maps each data-flow operator, with its inputs and outputs to modules, ports, and links. In doing so, the DAA avoids multiple assignments of hardware links… These last two mapping steps place the algorithmic description in a uniform notation for the expert analysis phase that follows." [Pg. 26] "The ISPS description is compiled into a VT data-flow representation, which makes it easier to recognize and implement design decisions by synthesis programs. The VT is a directed acyclic graph, DAG, similar in nature to those used in optimizing compilers, with the addition of control constructs to allow conditionals and subroutines in the VT. The nodes in this graph are called *operators* [i.e. actions] and correspond to operations that take certain values as input and produce new values as output… The arcs connecting the nodes are called *outnodes* and represent the generation or use of data values [i.e. conditions]. They are translations of the ISPS carriers and the temporary carriers needed to pass results from one operator to another. The graph is partitioned into subgraphs called *VT-bodies*, corresponding to a set of operations that can be evoked, entered, or left as a unit. These subgraphs are translations of ISPS procedures, labeled blocks and loops." [pg. 49]
selecting from said stored data for each of the specified definitions a corresponding integrated circuit hardware cell for performing the desired function of the application specific integrated circuit, said step of selecting a hardware cell comprising applying to the specified definition of the action or condition to be performed, a set of cell selection rules stored in said expert system knowledge base and generating for the selected integrated circuit hardware cells, a netlist defining the hardware cells which are needed to perform the desired function of the integrated	"*The rule interpreter component.* The rule interpreter pattern matches the working-memory elements against the rule memory to decide what rules apply to the given situation… The selection process of rules can be data driven, goal driven, or a combination of data and goal driven." [Pg. 12] "This section overviews allocating memories, registers, constants, controller, and database by recognizing certain features in the VT representation. These rules, like all the other rules in this chapter, use the service function rules to do their bookkeeping." [Pg. 69] "Once the designer has allocated the global non-changing hardware, the next task is to partition the whole design into smaller blocks and select a partition for allocation. Within each partition, designers allocate clock phase, operators, registers, data paths and control logic… The DAA allocates the clock phases, operators, registers, data paths and control logic in two subtasks, VT allocation and SCS allocation, which are shown in Figure 27. This allows the DAA to gather all the information about register usage in the VT allocation and then allocate registers and modules in the SCS allocation." [Pg. 71]

circuit and the interconnection requirements therefor	"If the control steps have not already been preallocated by VTDRIVE using the EMUCS algorithm, the DAA will create the maximally parallel non-pipeline design. The rules step through the VT operators sequentially, summing the delays specified for them in the technology-sensitive database." [Pg. 74] "*Register functions.* Next the designers allocate hardware operators, data paths and control logic. The DAA allocates operators either as register or ALU module attributes in the SCS representation." [Pg. 76] "The DAA compares the temporary register with registers that have been bound to hardware modules by size, cost estimation, and proximity estimation (See figure 16). It compares the temporary register to all bound registers of the same size and then to all bound registers of larger size. Next, ti compares the temporary register to all bound registers of a small size. If the cost estimation or the proximity estimation for the comparison is greater than the *cost* or *prox* attribute of the fold working-memory element for type REGISTER, then a goal of *module-mv* is created to combine the ports and attributes for the two register modules. Otherwise, the temporary register is bound to a new hardware register." [Pg. 81-82] "If a temporary ALU module is not removed by the rules in the previous section, then it must be combined with existing hardware or new hardware must be created. The DAA compares the temporary modules with ALUs that have been bound to hardware modules by type, size, cost estimation, and proximity estimation (See Figure 5)... If module attributes can be allocated, the temporary module is bound to a new hardware ALU." [Pg. 82-83]
14. A process as defined in claim 13, including generating from the netlist the mask data required to produce an integrated circuit having the desired function.	
15. A process as defined in claim 13 including the further step of generating data paths for the selected integrated circuit hardware cells.	"Within each partition, designers allocate clock phase, operators, registers, data paths and control logic... The DAA allocates the clock phases, operators, registers, data paths and control logic in two subtasks, VT allocation and SCS allocation, which are shown in Figure 27." [Pg. 71] "Chapter 5 has shown how designers partition the design task into four major tasks. These tasks involved global allocation of architectural modules and registers; local allocation of control steps, operators, registers, data paths and control; global allocations of operators and registers; and global improvements to the design." [Pg. 95]
16. A process as defined in claim 15 wherein said step of generating data paths comprises applying to the selected cells a set of data path rules stored in a knowledge base and generating the data paths therefrom.	See Table 14. RULES BY FUNCTION AND KNOWLEDGE TYPE [Pg. 63] "This section of rules encompasses much knowledge about making the least expensive connection from one port to another. Like the designer, they consider issues of existing data paths, expanding data paths, creating new data paths and multiplexors." [Pg. 66] "The link rules have knowledge about creating new links, using existing links, and expanding existing links." [Pg. 67]

17. A process as defined in claim 16 including the further step of generating control paths for the selected integrated circuit hardware cells.	

EXHIBIT B

Kowalski85c (DEF018108–DEF018118): Design Automation Assistant – CMU; AT&T Bell Labs
T.J. Kowalaski, et al., *The VLSI Design Automation Assistant: From Algorithms to Silicon*, IIEE Design & Test 33-43 (1985).

13. A computer-aided design process for designing an application specific integrated circuit which will perform a desired function comprising	
storing a set of definitions of architecture independent actions and conditions;	"We view hardware synthesis as the creation of a detailed representation from a more abstract representation. The designer first specifies the chip's functionality as an algorithm. The algorithmic description is not tied to a particular implementation style, such as parallel or serial, sequential or pipelined. Similarly, it is not restricted to a fabrication technology, such as TTL, ECL, NMOS, or CMOS." [Kowalski85c] at 34, DEF018109 – 2nd col. "The DAA actually works from a dataflow representation extracted from the ISPS description." [Kowalski85c] at 36. "The ISPS description is considered an algorithmic description, not a representation of the structure of the implementation." [Kowalski85c] at 47.
storing data describing a set of available integrated circuit hardware cells for performing the actions and conditions defined in the stored set;	"The DAA uses a technology-sensitive database… The technology-sensitive database contains expert knowledge about the trade-offs particular to the target technology." [pg. 34-35] "The technology-dependent modules are either selected from the module database or fabricated from simpler, equivalent database entries." [pg. 35] "The module binder chooses physically realizable, technology-dependent cells to implement the modules in the DAA's design. The cells are chosen from the module database according to user-supplied constraints on functionality, power dissipation, delay, and area. The module binder may, for example, choose a ripple-carry adder when area is the major design constraint, or a carry-lookahead adder when performing is the major design constraint." [Kowalski85c] at 37, DEF018112 – 1st col.
storing in an expert system knowledge base a set of rules for selecting hardware cells to perform the actions and conditions;	"A cornerstone of our hardware synthesis approach is the use of knowledge-based expert system. Such systems make decisions based on knowledge, expressed as rules, obtained from expert designers." [pg. 34] "The module binder: an algorithmic program that builds the modules specified by the DAA from components in the target technology. The technology-dependent modules are either selected from the module database or fabricated from simpler, equivalent database entries. The selection process is guided by user-supplied constraints on area, power, and speed. The module binder is the masters thesis project of E. Dirkes of Carnegie Mellon University." [Kowalski85c] at 35, DEF018110 – 1st col. "The DAA is a knowledge-based expert system that uses a database of over 500 rules to synthesize an architectural implementation from an algorithmic description with constraints." [Kowalski85c] at 36.
describing for a proposed application specific integrated circuit a series of architecture independent actions and conditions;	"We view hardware synthesis as the creation of a detailed representation from a more abstract representation. The designer first specifies the chip's functionality as an algorithm. The algorithmic description is not tied to a particular implementation style [i.e. architecture], such as parallel or serial, sequential or pipelined." [Kowalski85c] at 34. "The DAA is a knowledge-based expert system that uses a database of over 500 rules to synthesize an architectural implementation from an algorithmic description with constraints.

	The description is written in ISPS." [Kowalski85c] at 36. "The DAA actually works from a dataflow representation extracted from the ISPS description." [pg. 36] "The DAA produces a technology-independent hardware network. This network is composed of modules, ports, links, and symbolic microcode." [Kowalski85c] at 36 – 1st col.
specifying for each described action and condition of the series one of said stored definitions which corresponds to the desired action or condition to be performed; and	"The DAA actually works from a dataflow representation extracted from the ISPS description." [pg. 36] "The module binder: an algorithmic program that builds the modules specified by the DAA from components in the target technology… The selection process is guided by user-supplied constraints on area, power, and speed." [Kowalski85c] at 35 – 1st col.
selecting from said stored data for each of the specified definitions a corresponding integrated circuit hardware cell for performing the desired function of the application specific integrated circuit, said step of selecting a hardware cell comprising applying to the specified definition of the action or condition to be performed, a set of cell selection rules stored in said expert system knowledge base and generating for the selected integrated circuit hardware cells, a netlist defining the hardware cells which are needed to perform the desired function of the integrated circuit and the interconnection requirements therefor	"This algorithmic description is then transformed into a technology-independent network, built of modules of medium complexity … and a symbolic control description. This network is used to synthesize a technology-dependent network that includes implementations of the technology-independent modules and the controller in the target technology." [Kowalski85c] at 34. "A cornerstone of our hardware synthesis approach is the use of knowledge-based expert system. Such systems make decisions based on knowledge, expressed as rules, obtained from expert designers." [Kowalski85c] at 34. "DAA: a knowledge based expert system responsible for transforming the algorithmic description of the chip into a block diagram and a symbolic control specification. ... [T]he DAA uses a technology-sensitive database ...[which] contains expert knowledge about the trade-offs particular to the target technology. ... The module binder: an algorithmic program that builds the modules specified by the DAA from components in the target technology. The technology-dependent modules are either selected from the module database or fabricated from simpler, equivalent database entries." [Kowalski85c] at 34 – 35. "The module binder: an algorithmic program that builds the modules specified by the DAA from components in the target technology. The technology-dependent modules are either selected from the module database or fabricated from simpler, equivalent database entries. The selection process is guided by user-supplied constraints on area, power, and speed. The module binder is the masters thesis project of E. Dirkes of Carnegie Mellon University." [Kowalski85c] at 35, DEF018110 – 1st col. "This category also includes module generators – programs that produce related cells according to the designer's specifications. Module generators include . . . knowledge-based expert systems." [Kowalski85c] at 35 – 3rd col. "The module binder chooses physically realizable technology-dependent cells to implement the modules in the DAA's design. The cells are chosen from the module database according to the user-supplied constraints on functionality, power dissipation, delay, and area." [Kowalski85c] at 37 – 1st col. "The DAA is a knowledge-based expert system that uses a database of over 500 rules to synthesize an architectural implementation from an algorithmic description with constraints. The description is written in ISPS." [Kowalski85c] at 36. "The knowledge-based expert system approach followed by the DAA uses a weak method, match [i.e. an inference engine methodology], in place of extensive backtracking." [Kowalski85c] at 36.

	"The control allocator translates the technology-independent control specification, generated by the DAA, into a controller implementation in a chosen technology." [Kowalski85c] at 37. "The floor planner produces an abstract physical design for the technology-dependent network [i.e. netlist] designed by the DAA and the module binder." [Kowalski85c] at 37. "Figure 4. Sample netlist to floor plan transformation." [Kowalski85c] at 37.
14. A process as defined in claim 13, including generating from the netlist the mask data required to produce an integrated circuit having the desired function.	"The VLSI Design Automation Assistant: From Algorithms to Silicon" [Kowalski85c] at 33. "A series of programs translates an algorithm into chip design." [Kowalski85c] at 33. Fig. 1 [Kowalski85c] at 34. "The floor planner produces an abstract physical design for the technology-dependent network [a.k.a. netlist] designed by the DAA and the module binder. [Kowalski85c] at 37. "The data-path floor planner (DP): a knowledge-based expert system that is responsible for the complete physical design of the chip. It arranges the modules selected by the module binder and control allocator into a data-path floor plan and creates the cell-layout parameters". [Kowalski85c] at 35. "At the same time the floor plan is being built, the specifications for the modules – size, shape, delay, etc. – are being refined. These specifications will be passed to the module generators to guide the production of the cells." [Kowalski85c] at 38. "The generator program produces a Lava description of a cell from parameters that define the requirements on the cell." [Kowalski85c] at 39. "The Lava chip compiler compacts the individual cells and assembles them into a chip according to the floor plan… This symbolic layout can be macro-expanded into a pure layout that can be used to fabricate the chip." [Kowalski85c] at 40. "The floor planner produces an abstract physical design for the technology-dependent network designed by the DAA and the module binder." [Kowalski85c] at 37. "Figure 4. Sample netlist to floor plan transformation." [Kowalski85c] at 37.
15. A process as defined in claim 13 including the further step of generating data paths for the selected integrated circuit hardware cells.	"Our approach seeks to aid the designer with tools that automatically produce data paths and control sequences from an algorithmic system description within user-specified constraints". [Kowalski85c] at 33 "A cornerstone of our hardware synthesis approach is the use of knowledge-based expert systems. Such systems make decisions based on knowledge, expressed as rules, obtained from expert designers." [Kowalski85c] at 34. "The data-path floor planner (DP): a knowledge-based expert system that is responsible for the complete physical design of the chip. IT arranges the modules selected by the module binder and control allocator into a data-path floor plan and creates the cell-layout parameters." [Kowalski85c] at 35.
16. A process as defined in claim 15 wherein said step of generating data paths comprises applying to the selected cells a set of data path rules stored in a knowledge base and generating the data paths	"Our approach seeks to aid the designer with tools that automatically produce data paths and control sequences from an algorithmic system description within user-specified constraints". [Kowalski85c] at 33. "The DAA is a knowledge-based expert system that uses a database of over 500 rules to synthesize an architectural implementation from an algorithmic description with constraints. The description is written in ISPS" [Kowalski85c] at 36.

therefrom.	"The knowledge-based expert system approach followed by the DAA uses a weak method, match [an inference engine methodology], in place of extensive backtracking." [Kowalski85c] at 36. "The data-path floor planner (DP): a knowledge-based expert system that is responsible for the complete physical design of the chip. It arranges the modules selected by the module binder and control allocator into a data-path floor plan and creates the cell-layout parameters." [Kowalski85c] at 35. "The DP knowledge-based expert system is driven by a set of rules that embody expert knowledge about floor planning." [Kowalski85c] at 38 – 1st col.
17. A process as defined in claim 16 including the further step of generating control paths for the selected integrated circuit hardware cells.	"Our approach seeks to aid the designer with tools that automatically produce data paths and control sequences from an algorithmic system description within user-specified constraints". [Kowalski85c] at 33. "This algorithmic description is then transformed into a technology-independent network … and a symbolic control description. This network is used to synthesize a technology-dependent network that includes implementations of the technology-independent modules and the controller in the target technology." [Kowalski85c] at 34. "The control allocator: an algorithmic program that designs the controller for the data-path logic designed by the DAA…" [Kowalski85c] at 35. "The control allocator translates the technology-independent control specification, generated by the DAA, into a controller implementation in a chosen technology." [Kowalski85c] at 37. "The control allocator receives information from the technology-dependent hardware network and the module database [i.e. hardware cells] to produce an output file for a specific style of controller." [Kowalski85c] at 37. See Figure 1 [Kowalski85c] at 34.